Final Report for Contract # N00039-91-C-0162

Research on Wide Spectrum Languages and Research Environments for System Design and Specification

June 1, 1991 — September 30, 1995

Principal Investigator: David C. Luckham

1 Abstract

We designed new formal specification languages, tools and methodologies under this project. The languages, tools and methodologies allow the complete development process of large systems, from requirements and design through to testing and maintenance, to be subject to new analysis techniques based on machine processable formal specifications.

The formal specification languages used in the project or developed with full or partial support under this contract were Anna, TSL, TSL 1.5, VAL and Rapide. The analysis tools developed and enhanced as part of this project were the Anna Runtime Monitoring System and Anna Specification Analyzer. A number of methodologies and concepts based on formal specifications were developed as part of this project including debugging of specifications, methodologies for algebraic specification checking, methodologies for concurrent checking of specifications, methodologies for designing specifications and methodologies for system maintenance using specifications. A number of papers and technical reports and one book was published under this contract. A list of publications is included in this report.

A lot of the effort in the project was spent on technology transfer. The training activities on formal specification included university courses and a tutorial in TriAda '91. Anna tools were ported to a number of platforms and the subset of Ada handled was enhanced. Anna was and is being used at a number of sites outside Stanford.

2 Introduction

The objective of this project was the development of new specification languages and techniques for the cost effective production of highly reliable, reusable Ada

components.

Our approach was to design a specification language based on Anna and to develop environment tools supporting applications of this language to Ada and other languages. The specification language and tools facilitate new development processes for large Ada systems. These processes allow all stages of the life-cycle from requirements analysis and design through to testing and maintenance, to be subjected to new automated analysis and synthesis techniques based on machine processable formal specifications.

We also utilized Anna technology to develop a specification language component of Rapide. This involves designing a specification language suitable for constraint-based specification of distributed systems that is both powerful and easily checkable at runtime. This version of the specification language is being designed to apply to the poset (partially ordered event set) executions of Rapide, and also to include temporal operators.

Much of our current effort is devoted to technology transfer by improving our Anna toolset. Development of new tools focuses on implementation algorithms to take advantage of new advances in high performance computing. Improvements in the Anna toolset emphasizes support for new rigorous methods of applying formal specifications, and also is in response to feedback from the Anna user community.

3 Results Achieved

3.1 Specification Languages

A number of specification languages were developed and enhanced as part of this project. TSL was extended to TSL 1.5 [30] and the ideas from Anna, TSL and VAL were used in the development of Rapide [27, 50].

3.2 Specification-based Tools

3.2.1 The Anna Specification Analyzer

We extended the Anna toolset with a new tool, the Anna Package Specification Analyzer, that supports requirements analysis by symbolic execution and logical inference applied to package interface specifications. This is an advanced tool for analyzing formally specified package interfaces. It allows designers to check conformance of the Ada interfaces with requirements prior to implementation. The tool and its uses are documented in a user's guide available as Stanford University Technical Note CSL-TN-93-390.

3.2.2 Anna Runtime Checking System

The Anna runtime checking tools have been extended to support full Ada'83 as well as many more Anna constructs such as algebraic specifications. We upgraded to a new front-end generated by Arcadia's Aflex and Ayacc. The full Anna toolset supports early life-cycle activities as well as testing and debugging. Significant new tool features include checking of algebraic specifications and using multiprocessor platforms to speed-up the checking process.

Runtime Checking of Algebraic Specifications The methodology developed for algebraic specification checking was implemented and incorporated into the run-time checking tools. The methodology involves an incremental theorem prover that monitors the execution of the underlying program. The theorem prover determines when certain checks need to be performed on values generated by the program. Programmers can now have a reasonable portion of their axiomatic specifications of Ada packages checked at run-time. The details of this methodology is described in a paper published in TAV'91.

Concurrent Runtime Checking on Multi-Processors Concurrent checking of annotations was also implemented in the Anna toolset and was released. This allows the runtime overhead of annotation checking to be off-loaded to other processors.

A New Graphical User Interface A graphical user interface to the Anna Debugging system was implemented using the Chiron tools available from the ARPA Arcadia project at UC Irvine. This interface creates and maintains several windows corresponding to various requirements of Anna debugging: program source, program I/O, source of which annotations were violated, and source of where the annotations were violated. The interface was demonstrated at InterCHI'93, the 1993 Conference on Human Factors in Computing Systems, Amsterdam, The Netherlands, April 24-29, 1993, sponsored by ACM/SIGCHI.

3.3 New Methodologies, Concepts and Ideas

3.3.1 Extension of Anna Runtime Checking Technology to Algebraic Specifications

Run-time checking of formal specifications was extended to more complex Anna constructs such as algebraic specifications. Algebraic specification checking requires maintaining program history as well as theorem proving at run-time. A paper describing this algorithm was published in TAV'91.

3.3.2 Building Secure Systems using Anna and Concurrent Checking

A new methodology was developed for building secure Ada systems using runtime checking of formal Anna specifications on high performance multiprocessors. Programs are monitored continuously for specification consistency during program execution; the time penalty incurred in checking is minimized by distributing monitoring onto different processors. Permanent runtime monitoring is aimed particularly at security and safety problems that occur even in systems produced by the most rigorous formal methods, for example due to failures in hardware, compiler errors, etc. This methodology is described in a paper published in IEEE Computer, March 1993.

3.3.3 Debugging using Formal Specifications

New techniques for applying Anna to Ada software production were developed. A debugging technique called "2-dimensional pinpointing" was published in IEEE Software, Jan.'91.

3.3.4 Specification Development Methodologies

Specification development methodologies were published in COMPSAC'90.

3.3.5 Software Maintenance using Formal Specifications

Software maintenance and reuse techniques using formal specifications have been published in CSM'90.

3.3.6 Application of Anna Technology to Other Languages

A study was conducted for designing an Anna-like language for C++.

Anna concepts and technology have been reused extensively in the development of the APP system by Dr. D. Rosenblum at ATT Bell Labs. APP is an Annotation Preprocessor for C programs developed in Unix-based environments. APP allows annotations to be used to detect a wide variety of faults in C programs. It is designed and engineered for ease-of-use in industry. APP has been used in the development of a variety of software systems in C over the past few years.

3.3.7 Prototech Technology Transfer

We adapted TSL, VAL and Anna technology into our Rapide Prototech environment. This work involved extending and combining TSL, VAL and Anna into a

language for specifying and checking the behavior of Rapide prototypes. Annotations are based not only on predicates (as in Anna) but also on patterns of events (as in TSL) that express constraints on partial-orders of events and temporal relationships between events in concurrent Rapide program executions.

3.3.8 Type checking in the presence of specifications

We investigated algorithms to extend the standard static type checking of Rapide to include semantic information based on formal specifications in the new specification language.

3.4 Technology Transfer

3.4.1 Courses, Tutorials and Other Educational Activities

We conducted a full-day tutorial course on Anna and formal methods during TriAda'91, and have taught Anna to students at Stanford SUNY Binghamton and UC, Irvine. In addition, we published a book on Anna programming methodology and on Anna program examples. In addition, we published many other reports to expose computer professional to formal methods technology using Anna and Ada. A selection of published articles is included below. The Anna tools have also been upgraded to support full Ada'83 and has been distributed for use in various training programs.

3.4.2 Applications of Anna and Tools Outside PAVG

Anna technology transfer was streamlined due to increased demand. We maintain an electronic-mail response service, and use FTP to allow quick acquisition of the tools. The toolset has been acquired by at least 100 parties worldwide. The source code is also made available to parties interested in porting to other platforms such as Sparcstations, VAXes, and 386-based PC's. Current uses include large-scale systems testing, graduate school courses, and applications of components of the tool suite.

Listed below are a few outside Anna users and a brief description of how Anna was and is being used:

- Robert Willis of Hampton University developed a series of very large Ada projects and used Anna in their development.
- Rodney L. Bown of University of Houston, Clear Lake ported Anna for use in mission critical NASA software projects.

- Allan Willey of Motorola used the toolset in the Cellular Infrastructure development group for Ada projects.
- Rod Chapman of York University, England integrated the Anna toolset and York University Ada compiler. This included extending and modifying the Anna language.
- Fernando Naveda of University of Scranton used the Anna tools in a graduate course for teaching good software development practices.
- Reg Meeson of IDA used the toolset as a technology demonstration/evaluation exercise.
- Francois Charlot of Universite Paris-Sud worked on a PhD thesis on specifying large software systems and used Anna as an abstraction language.
- Sergio Antoy of Portland State University used the Anna toolset in graduate courses on formal methods for over three years.
- Richard Taylor of University of California, Irvine has taught Anna and used the toolset in an undergraduate course on software engineering.
- Larry Matthias of Lockheed Engineering and Sciences Company, Hampton, Virginia, used the Anna toolset in a project for NASA involving an image processing system for satellite data.

Technology transfer is being actively promoted, not only by upgrading the Anna toolset to support full Anna and Ada'83, but also porting to a variety of computers. The toolset source is available via anonymous FTP. Ports of Anna tools to other platforms are also being made by Anna users.

3.4.3 Anna use in Design of DSSA Languages

IBM DSSA team presentations at the ARPA DSSA March 1993 meeting document the use of LileAnna, which reuses Anna as a subset, in the process of developing the IBM ADAGE Avionics architecture. Anna and LileAnna were also reviewed in the Honeywell DSSA team presentation by Dr. S. Vestal at the same meeting.

3.4.4 Reuse of Anna at ATT for Checking C Programs

Anna concepts and technology have been reused extensively in the development of the APP system by Dr. D. Rosenblum at ATT Bell Labs. APP is an Annotation Preprocessor for C programs developed in Unix-based environments. APP allows annotations to be used to detect a wide variety of faults in C programs. It is

designed and engineered for ease-of-use in industry. APP has been used in the development of a variety of software systems in C over the past few years.

Additionally, Anna was added to the NASA's AdaNET public domain repository.

3.4.5 Availablity of Anna on Multiple Platforms

The Anna toolset has been ported to the following platforms: Vax/VMS, Vax/Ultrix, SparcStation, Meridian compiler, York Ada compiler, SCO UNIX/386, MSDOS/386, IBM RS/6000.

The most recent Anna toolset release includes the Specification Analyzer, concurrent runtime specification checking, runtime checking of algebraic axioms, and the Arcadia Chiron/Anna interface. The full toolset source has been made available via anonymous FTP. Implementation-dependent portions of the toolset have been isolated to a single package, and alternate implementations of this package are being maintained for Sun/3, SparcStation 2, VAX/VMS, and Sequent Symmetry. The toolset has been distributed to at least sixty sites: twenty in academia and forty in industry.

3.5 Papers and Books

A. Goyal and S. Sankar. *The Application of Formal Specifications to Software Documentation and Debugging.* Technical Report 93-392, Computer Systems Laboratory, Stanford University, 1993.

Sriram Sankar and Anoop Goyal and Prakash Sikchi. *Software Testing using Algebraic Specification Based Test Oracles.* Technical Report 93-566, Computer Systems Laboratory, Stanford University, 1993.

W. Mann. *The anna package specification analyzer user's guide.* Technical Note CSL-TN-93-390, Computer Systems Lab, Stanford University, January 1993.

S. Sankar and M. Mandal. *Concurrent runtime monitoring of formally specified programs.* Technical Report 90-425, Computer Systems Laboratory, Stanford University, 1990. Also published in IEEE Computer 1993.

D. C. Luckham, S. Sankar, W. Mann, and A. Goyal. *Anna - A Language for Specifying Ada.* In Proceedings of the DARPA Software Technology Conference. Los Angeles, California. April, 1992. Pages 498–501.

S. Sankar. *Run-time consistency checking of algebraic specifications.* In Proceedings of the Symposium on Testing, Analysis, and Verification (TAV4), pages 123-129, Victoria, Canada, October 1991. ACM Press.

J. J. Kenney and W. Mann. *Anna Package Specification: Case Studies.* Technical Report 91-496, Computer Systems Laboratory, Stanford University, October 1991. (Program Analysis and Verification Group Report 56).

M. Walicki, J. U. Skakkebaek, and S. Sankar. *The Stanford Ada style checker: An application of the Anna tools and methodology.* Technical Report 91-488, Computer Systems Laboratory, Stanford University, August 1991. (Program Analysis and Verification Group Report 55).

S. Sankar and D. S. Rosenblum. *Runtime Checking and Debugging of Formally Specified Programs.* Surveyors' Forum, ACM Computing Surveys. 23(1) March, 1991. Pages 125–127.

P. R. H. Place, W. G. Wood, D. C. Luckham, W. Mann, and S. Sankar. *Formal development of Ada programs using Z and Anna: A case study.* Technical Report CMU/SEI-91-TR-1, Software Engineering Institute, Carnegie-Mellon University, February 1991.

D. C. Luckham, S. Sankar, and S. Takahashi. *Two dimensional pinpointing: An application of formal specification to debugging packages.* IEEE Software, 8(1):74-84, January 1991. (Also Stanford University Technical Report No. CSL-TR-89-379.).

D. C. Luckham. *Programming with Specifications: An Introduction to ANNA, A Language for Specifying Ada Programs.* Texts and Monographs in Computer Science. Springer-Verlag, October, 1990.

S. Sankar and M. Mandal. *Concurrent runtime monitoring of formally specified programs.* IEEE Computer, March 1993. Pages 32-41.

D. C. Luckham, S. Sankar, W. Mann, and A. Goyal. *Anna - a language for specifying ada.* In Proceedings of the DARPA Software Technology Conference. Los Angeles, California. April, 1992. Pages 498-501.

D. Rosenblum, *A Practical Approach to Programming with Assertions*, forthcoming, IEEE Transactions on Software Engineering.

4 Conclusions

References

[1] L. M. Augustin, B. A. Gennart, Y. Huh, D. C. Luckham, and A. G. Stanculescu. Verification of VHDL designs using VAL. In *Proceedings of the 25th Design Automation Conference (DAC)*, pages 48–53, Anaheim, CA, June 1988. IEEE Computer Society Press.

[2] Larry M. Augustin, David C. Luckham, Benoit A. Gennart, Youm Huh, and Alec G. Stanculescu. *Hardware Design and Simulation in VAL/VHDL.* Kluwer Academic Publishers, October 1990. 322 pages.

[3] Doug Bryan. Rapide–0.2 language and tool-set overview. Technical Note CSL–TN–92–387, Computer Systems Lab, Stanford University, February 1992.

[4] Doug Bryan. Using rapide to model and specify inter-object beahvior. In *OOPSLA '94 workshop on Precise behavioral specifications in OO information modeling*, Oct. 24, 1994.

[5] Douglas L. Bryan. Run-time monitoring of tasking behavior using a specification language. In *Second Workshop on Large Grain Parallelism*, Pittsburgh, Pennsylvania, 11–14 October 1987. Software Engineering Institute, Carnegie Mellon University. An extended abstract.

[6] Douglas L. Bryan. An algebraic specification of the partial orders generated by concurrent Ada computations. In *Proceedings of Tri-Ada '89*, pages 225–241. ACM Press, October 1989.

[7] Anoop Goyal and Sriram Sankar. Software documentation and testing using formal specifications: An application of anna and its associated tool-set. In preparation.

[8] D. P. Helmbold. The meaning of TSL: An abstract implementation of TSL-1. Technical Report CSL–TR–88–353, Computer Systems Laboratory, Stanford University, March 1988. Also published by Computer Information Sciences Board, UC Santa Cruz as UCSC–CRL–87–29.

[9] D. P. Helmbold and D. C. Luckham. Runtime detection and description of deadness errors in Ada tasking. Technical Report 83-249, Computer Systems Laboratory, Stanford University, November 1983. (Program Analysis and Verification Group Report 22).

[10] D. P. Helmbold and D. C. Luckham. Debugging Ada tasking programs. *IEEE Software*, 2(2):47–57, March 1985. (Also Stanford University Computer Systems Laboratory Technical Report No. 84-262).

[11] D. P. Helmbold and D. C. Luckham. TSL: Task sequencing language. In *Ada in Use: Proceedings of the Ada International Conference*, pages 255–274. Cambridge University Press, May 1985.

[12] David P. Helmbold and Douglas L. Bryan. Design of run time monitors for concurrent programs. Technical Report CSL–TR–89–395, Computer Systems Laboratory, Stanford University, October 1989.

[13] Alexander Hsieh. Rapide-0.2 examples. Technical Report CSL–TR–92–510, Computer Systems Lab, Stanford University, February 1992.

[14] D. Katiyar, D. C. Luckham, N. Madhav, S. Meldal, J. C. Mitchell, and S. Sankar. Subtyping, assignment, and cloning in a concurrent object-oriented language. In *Proceedings of the DARPA Software Technology Conference*, pages 458–470, Los Angeles, California, April 1992. DARPA.

[15] John J. Kenney. *Executable Formal Models of Distributed Transaction Systems based on Event Processing.* PhD thesis, Stanford University, December 1995. forthcoming Stanford Dissertation.

[16] B. Krieg-Brückner. Consistency checking in Ada and Anna: A transformational approach. *Ada Letters*, 3(2):46–54, September-October 1983.

[17] B. Krieg-Brückner and D. C. Luckham. Anna: Towards a language for annotating Ada programs. *ACM SIGPLAN Notices*, 15(11):128–138, 1980.

[18] D. C. Luckham, D. P. Helmbold, S. Meldal, D. L. Bryan, and M. A. Haberler. Task sequencing language for specifying distributed Ada systems: TSL–1. In Habermann and Montanari, editors, *System Development and Ada, proceedings of the CRAI workshop on Software Factories and Ada. Lecture Notes in Computer Science. Number 275*, pages 249–305. Springer-Verlag, May 1986.

[19] D. C. Luckham, S. Sankar, W. Mann, and A. Goyal. Anna — a language for specifying Ada. In *Proceedings of the DARPA Software Technology Conference*, pages 498–501, Los Angeles, California, April 1992. DARPA.

[20] D. C. Luckham, S. Sankar, and S. Takahashi. Two dimensional pinpointing: An application of formal specification to debugging packages. *IEEE Software*, 8(1):74–84, January 1991. (Also Stanford University Technical Report No. CSL–TR–89–379.).

[21] D. C. Luckham and F. W. von Henke. An overview of Anna, a specification language for Ada. *IEEE Software*, 2(2):9–23, March 1985.

[22] David C. Luckham. *Programming with Specifications: An Introduction to ANNA, A Language for Specifying Ada Programs.* Texts and Monographs in Computer Science. Springer-Verlag, October, 1990.

[23] David C. Luckham and Benoit A. Gennart. Event patterns: a language construct for hierarchical design of concurrent systems.

[24] David C. Luckham, David P. Helmbold, Sigurd Meldal, Douglas L. Bryan, and Michael A. Haberler. Task sequencing language for specifying distributed Ada systems: TSL-1. In *Proceedings of PARLE: Conference on Parallel Architectures and Languages Europe. Lecture Notes in Computer Science. Number 259, Volume II: Parallel Languages*, pages 444–463, Eindhoven, The Netherlands, 15–19 June 1987. Springer-Verlag.

[25] David C. Luckham, Randall B. Neff, and David S. Rosenblum. An environment for Ada software development based on formal specification. Technical Report CSL-TR-86-305, Stanford University, August 1986. Also published in as an article in Ada Letters, VII(3):94–106, May/June 1987.

[26] David C. Luckham and James Vera. An event-based architecture definition language. *IEEE Transactions on Software Engineering*, 21(9):717–734, September 1995.

[27] David C. Luckham, James Vera, Doug Bryan, Larry Augustin, and Frank Belz. Partial orderings of event sets and their application to prototyping concurrent, timed systems. *Journal of Systems and Software*, 21(3):253–265, June 1993.

[28] David C. Luckham, James Vera, and Sigurd Meldal. Three concepts of system architecture. *submitted to the Communications of the ACM*, July 1995.

[29] David C. Luckham, Friedrich W. von Henke, Bernd Krieg-Brückner, and Olaf Owe. *ANNA, A Language for Annotating Ada Programs*, volume 260 of *Lecture Notes in Computer Science*. Springer-Verlag, 1987.

[30] D.C. Luckham, S. Meldal, D.P. Helmbold, D.L. Bryan, and W. Mann. An introduction to Task Sequencing Language, TSL 1.5. Technical Report 38, Department of Informatics, University of Bergen, Bergen, Norway, August 1989. Preliminary version.

[31] N. Madhav and W. R. Mann. A methodology for formal specification and implementation of Ada packages using Anna. In *Proceedings of the Computer Software and Applications Conference, 1990*, pages 491–496. IEEE Computer Society Press, 1990. (Also Stanford University Computer Systems Laboratory Technical Report No. 90-438).

[32] N. Madhav and S. Sankar. Application of formal specification to software maintenance. In *Proceedings of the Conference on Software Maintenance*, pages 230–241. IEEE Computer Society Press, November 1990.

[33] Neel Madhav. An Ada-Prolog system. In *International Conference on Computing and Information*, pages 340–344, Niagara Falls, Canada, May 1990. (Also Stanford University, Computer Systems Lab technical report CSL-TR-90-437. PAVG technical Report No. 49).

[34] W. Mann, J. Kenney, and S. Sankar. Prototype semantic analyzers for rapidly changing languages. Program Analysis and Verification Group internal document, Computer Systems Lab, Stanford University, September 1992.

[35] Walter Mann. The Anna package specification analyzer user's guide. Technical Note CSL–TN–93–390, Computer Systems Lab, Stanford University, January 1993.

[36] Sigurd Meldal, Sriram Sankar, and James Vera. Exploiting locality in maintaining potential causality. In *Proceedings of the Tenth Annual ACM Symposium on Principles of Distributed Computing*, pages 231–239, New York, NY, August 1991. ACM Press. Also Stanford University Computer Systems Laboratory Technical Report No. CSL–TR–91–466.

[37] R. Neff. *Ada/Anna Package Specification Analysis.* PhD thesis, Stanford University, December 1989. Also Stanford University Computer Systems Laboratory Technical Report No. CSL–TR–89–406.

[38] P. R. H. Place, W. G. Wood, D. C. Luckham, W. Mann, and S. Sankar. Formal development of Ada programs using Z and Anna: A case study. Technical Report CMU/SEI-91-TR-1, Software Engineering Institute, Carnegie-Mellon University, February 1991.

[39] D. S. Rosenblum. *Design and Verification of Distributed Tasking Supervisors for Concurrent Programming Languages.* PhD thesis, Stanford University, March 1988. Also Stanford University Computer Systems Laboratory Technical Report No. CSL–TR–88–357.

[40] D. S. Rosenblum, S. Sankar, and D. C. Luckham. Concurrent runtime checking of annotated Ada programs. In *Proceedings of the 6th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 10–35. Springer-Verlag — Lecture Notes in Computer Science No. 241, December 1986. (Also Stanford University Computer Systems Laboratory Technical Report No. 86-312).

[41] David S. Rosenblum. A methodology for the design of Ada transformation tools in a DIANA environment. Technical Report 85-269, Stanford University, February 1985. Also published in IEEE Software, 2(2):24–33, March 1985.

[42] David S. Rosenblum. Specifying concurrent systems with TSL. *IEEE Software*, 8(3):52–61, May 1991.

[43] S. Sankar. *Automatic Runtime Consistency Checking and Debugging of Formally Specified Programs.* PhD thesis, Stanford University, August 1989. Also Stanford University Department of Computer Science Technical Report No. STAN–CS–89–1282, and Computer Systems Laboratory Technical Report No. CSL–TR–89–391.

[44] S. Sankar. A note on the detection of an Ada compiler bug while debugging an Anna program. *ACM SIGPLAN Notices*, 24(6):23–31, 1989.

[45] S. Sankar and M. Mandal. Concurrent runtime monitoring of formally specified programs. Technical Report 90–425, Computer Systems Laboratory, Stanford University, 1990. Also in IEEE Computer, March 1993.

[46] S. Sankar and D. S. Rosenblum. The complete transformation methodology for sequential runtime checking of an Anna subset. Technical Report 86-301, Computer Systems Laboratory, Stanford University, June 1986. (Program Analysis and Verification Group Report 30).

[47] S. Sankar, D. S. Rosenblum, and R. B. Neff. An implementation of Anna. In *Ada in Use: Proceedings of the Ada International Conference, Paris*, pages 285–296. Cambridge University Press, May 1985.

[48] Sriram Sankar. Run-time consistency checking of algebraic specifications. In *Proceedings of the Symposium on Testing, Analysis, and Verification (TAV4)*, pages 123–129, Victoria, Canada, October 1991. ACM Press.

[49] Sriram Sankar, Anoop Goyal, and Prakash Sikchi. Software testing using algebraic specification based test oracles. Forthcoming Stanford University Technical Report, December 1993.

[50] Rapide Design Team. *The Rapide-1 Full Syntax Reference Manual.* Program Analysis and Verification Group, Computer Systems Lab., Stanford University, version 1 edition, August 1995.

[51] F. W. von Henke, D. C. Luckham, B. Krieg-Brückner, and O. Owe. Semantic specification of Ada packages. In *Ada in Use: Proceedings of the Ada International Conference*, pages 185–196. Cambridge University Press, May 1985.

[52] M. Walicki, J. U. Skakkebaek, and S. Sankar. The stanford Ada style checker: An application of the Anna tools and methodology. Technical Report 91-488. Computer Systems Laboratory, Stanford University, August 1991. (Program Analysis and Verification Group Report 55).